Radial Basis Functions for High-Dimensional Visualization

Vaclav Skala
Department of Computer Science and Engineering
University of West Bohemia
CZ 306 14 Plzen, Czech Republic
skala@kiv.zcu.cz

***Abstract* —** High-dimensional visualization is usually connected with large data processing. Because of dimensionality, it is nearly impossible to make a tessellation, like the Delaunay tessellation in E^d, followed by data interpolation. One possibility of data interpolation is the use of the Radial Basis Functions (RBF) interpolation. The RBF interpolation supports the interpolation of scattered data in *d*-dimensional space. The computational cost of the RBF interpolation is higher but does not increase significantly with the data dimensionality. It increases with the number of values to be processed non-linearly. In this paper, the RBF interpolation properties will be discussed as well as how to process data incrementally. Incremental computation decreases computational complexity and decreases RBF computational cost for the given data set significantly, especially for the visualization purposes, when the interpolated/approximated data are used many times. As the proposed approach is based on a solution of a system of linear equations, the RBF interpolation is convenient especially for data sets processing using matrix-vector or GPU architectures.

Keywords - Visualization; computer graphics; interpolation; radial basis functions; RBF

I. INTRODUCTION

Visualization of potential (scalar) fields in a multi-dimensional space is a typical problem not only in physical sciences. The problem seems to be quite simple, but it is actually a quite complicated task. In the E^2 case the usual approach is to tessellate the domain (e.g. *x-y* space) and then to use linear interpolation or cubic interpolation. In general, the computational complexity of the Delaunay tessellation (DT) for *N* points in the *d*-dimensional case is of $O(d\,N^2)$ complexity. It needs to be noted that the DT is not easy to implement in the the *d*-dimensional space. There is also a severe problem how to smoothly interpolate scalar values in the *d*-dimensional space. The vast majority of interpolation techniques rely on "separable" interpolations, i.e. interpolation is made in each axis independently expecting that the selection of axes order is arbitrary. Unfortunately such approaches lead to some artifacts and caused errors are unpredictable.

Radial basis function (RBF) interpolation belongs to non-separable interpolations used for interpolation in *d*-dimensional space. The computational cost of RBF increases non-linearly with the number of data processed and linearly with the dimensionality of the data set. The RBF interpolation is based on a distance of two points, i.e. the distance of two points $r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ is computed. The great advantage of RBF interpolation is that it does not need any tessellation of the data domain and simply supports the data of any dimensionality. RBF applications are quite widespread and can be found in data visualization, solutions of partial differential equations (PDE), neural networks, reconstruction of corrupted images etc.

The computational cost of the RBF interpolation is higher as the cost of tessellation is inheritably covered into the RBF interpolation in principle. Two significant aspects are connected with the RBF:

- re-computation of the RBF interpolation and
- reduction of the data set.

It should be noted that the RBF interpolation leads to a solution of linear system of equations (LSE) $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. The proposed approaches are valid for the *d*-dimensional case, but in the following text, $d = 2$ will be used for explanation.

II. RADIAL BASIS FUNCTION INTERPOLATION

RBF interpolation is quite simple from a mathematical point of view. It is based on a distance computing of two points in the *d*-dimensional space. RBF interpolation is defined by the function:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j\, \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) = \sum_{j=1}^{N} \lambda_j\, \varphi_j(r_j)$$
$$r_j = \|\boldsymbol{x} - \boldsymbol{x}_j\|$$

It means that for the given data set $\{< x_i, h_i >\}_{i=1}^{N}$, where h_i are associated values to be interpolated and $\boldsymbol{x}_i$ are domain coordinates, a linear system of equations is obtained:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{N} \lambda_j\, \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) \qquad i = 1, \dots, N$$

where λ_j are weights to be computed. Due to stability issues, usually a polynomial $P_k(\boldsymbol{x})$ of a degree *k* is added to the form, i.e.:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{N} \lambda_j\, \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) + P_k(\boldsymbol{x}_i) \qquad i = 1, \dots, N$$

 ISBN: 978-1-61208-184-7

For a practical reason in many applications, the polynomial of the 1st degree is used, i.e. linear polynomial $P_1(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + a_0$. Then the RBF interpolation function has the following form:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{N} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right) + \boldsymbol{a}^T\boldsymbol{x}_i + a_0$$

$$h_i = f(\boldsymbol{x}_i) \qquad i = 1, \dots, N$$

and additional conditions are applied:

$$\sum_{j=1}^{N} \lambda_i = 0 \qquad \sum_{j=1}^{N} \lambda_i \boldsymbol{x}_i = \boldsymbol{0}$$

For the *d*-dimensional case and *N* points given, a system of $(N + d + 1)$ linear equations has to be solved.

For *d=2* vectors $\boldsymbol{x}_i$ and $\boldsymbol{a}$ are given as $\boldsymbol{x}_i = [x_i, y_i]^T$ and $\boldsymbol{a} = [a_x, a_y]^T$. Using the matrix notation we can write:

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,N} & x_1 & y_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \varphi_{N,1} & \cdots & \varphi_{N,N} & x_N & y_N & 1 \\ x_1 & \cdots & x_N & 0 & 0 & 0 \\ y_1 & \cdots & y_N & 0 & 0 & 0 \\ 1 & \cdots & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ a_x \\ a_y \\ a_0 \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_N \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix} \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\boldsymbol{a}^T\boldsymbol{x}_i + a_0 = a_x x_i + a_y y_i + a_0$$

It can be seen that for the 2-dimensional case and *N* points given a system of $(N + 3)$ linear equations has to be solved. It can be seen that the RBF interpolations are not "separable" by the definition, i.e. an interpolation over x-axis and then over y-axis and vice versa cannot be made.

The radial basis functions interpolation was originally introduced using multiquadric method [5] in 1971 called Radial Basis Function method. Since then, many different RBF interpolation schemes have been developed with some specific properties, e.g. [4] uses $\varphi(r) = r^2 lg\, r$, which is called Thin-Plate Spline (TPS), a function $\varphi(r) = e^{-(\epsilon r)^2}$ that was proposed in [9]. Compactly Supported RBF (CSRBF) was introduced in [13] as

$$\varphi(r) = \begin{cases} (1-r)^q \, P(r), & 0 \le r \le 1 \\ 0\,, & r > 1 \end{cases},$$

where: $P(r)$ is a polynomial function and *q* is a parameter.

Theoretical problems with stability and solvability were resolved by [6] and [14]. Generally, there are two main groups of the RBFs:

- "global" – a typical example is TPS function
- "local" – Compactly supported RBF (CSRBF)

If the "global" functions are taken, the matrix $\boldsymbol{A}$ of the LSE is full, for large *N* is becoming ill-conditioned and problems with convergence can be expected. On the other hand if the CSRBFs are taken, the matrix $\boldsymbol{A}$ is becoming relatively sparse, i.e. computation of the LSE will be faster, but the scaling factor needs to be carefully selected due to a limited influence of the CSRBF and the final function tends to be "blobby" shaped.

TABLE I. TYPICAL EXAMPLE OF "GLOBAL" FUNCTIONS

"Global" functions	$\phi(r)$
Thin-Plate Spline (TPS)	$r^2 \log r$
Gauss function	$\exp\left(-(\varepsilon r)^2\right)$
Inverse Quadric (IQ)	$1/\left(1+(\varepsilon r)^2\right)$
Inverse multiquadric (IMQ)	$1/\sqrt{1+(\varepsilon r)^2}$
Multiquadric (MQ)	$\sqrt{1+(\varepsilon r)^2}$

TABLE II. TYPICAL EXAMPLE OF "LOCAL" CSRBF FUNCTIONS

ID	**Function**
1	$(1-r)_+$
2	$(1-r)_+^3(3r+1)$
3	$(1-r)_+^5(8r^2+5r+1)$
4	$(1-r)_+^2$
5	$(1-r)_+^4(4r+1)$
6	$(1-r)_+^6(35r^2+18r+3)$
7	$(1-r)_+^8(32r^3+25r^2+8r+1)$
8	$(1-r)_+^3$
9	$(1-r)_+^3(5r+1)$
10	$(1-r)_+^7(16r^2+7r+1)$

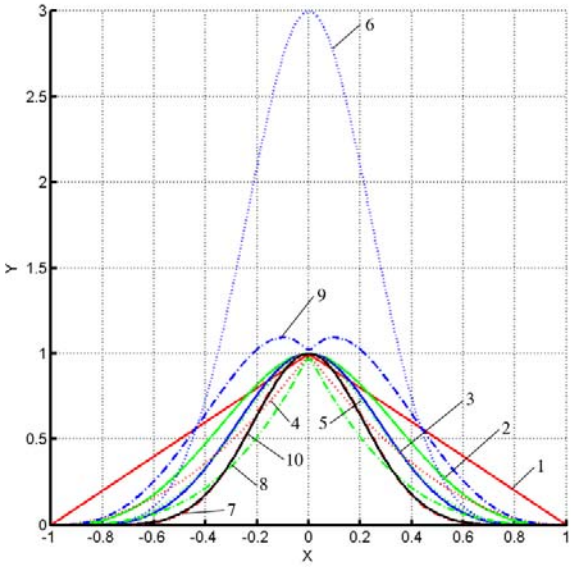


Figure 1. Geometrical properties of CSRBF

Tab. 2 presents typical examples of CSRBFs defined for the interval < 0 , 1 >, but for the practical use a scaling is used, i.e. the value *r* is multiplied by a scaling factor α, where 0 < α < 1. Fig. 1 presents the geometrical properties of typical CSRBFs.

III. INCREMENTAL RBF COMPUTATION

Some interesting problems can be solved using RBF interpolation quite effectively, e.g. surface reconstruction from scattered data [3][8][9][16], reconstruction of damaged images [11][15], inpainting removal [2][12] etc. All those applications based on RBFs interpolation have one significant disadvantage – the computational cost. This is especially severe in applications when the data are not static. Typical examples of non-static data are:

- Position of points has changed. It means that the whole system of linear equations has to be formed and recomputed which leads generally to $O(N^3)$ computational complexity and unacceptable time-consuming computation.
- Position of points remains fixed, but the value associated with a point has changed. In this case, iterative methods are usually faster than explicit computation of an inverse matrix.

In some applications a "sliding window" on data is required, especially in time-related applications when old data should not be used in the interpolation and new data are to be included. This is a typical situation in signal processing applications. Considering the above facts above there is a question how to compute RBF incrementally with a lower computational complexity.

The main question to be answered is:

Is it possible to use already computed RFB interpolation if a new point is to be included to the data set?

If the answer is positive it should lead to significant decrease of computational complexity. In the following, it will be presented how a new point can be inserted, how a selected point can be removed and also how to select the best candidate for a removal according to an error caused by this point removal.

Let us consider some operations with block matrices (assuming that all operations are correct and matrices are non-singular in general etc.).

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \\ -(\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & (\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \end{bmatrix}$$

Let us consider a matrix $\boldsymbol{M}$ of $(n+1) \times (n+1)$ and a matrix $\boldsymbol{A}$ of $n \times n$ in the following block form:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

Then the inverse of the matrix $\boldsymbol{M}$ applying the rule above can be written as:

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \left(\boldsymbol{A}-\frac{1}{c}\boldsymbol{b}\boldsymbol{b}^T\right)^{-1} & -\frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b} \\ -\frac{1}{k}\boldsymbol{b}^T\boldsymbol{A}^{-1} & \frac{1}{k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^{-1}+\frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{A}^{-1} & -\frac{1}{k}\boldsymbol{A}^{-1}\boldsymbol{b} \\ -\frac{1}{k}\boldsymbol{b}^T\boldsymbol{A}^{-1} & \frac{1}{k} \end{bmatrix}$$

where: $k = c - \boldsymbol{b}^T\boldsymbol{A}^{-1}\boldsymbol{b}$

We can easily simplify this equation if the matrix $\boldsymbol{A}$ is symmetrical as:

$$\boldsymbol{\xi} = \boldsymbol{A}^{-1}\boldsymbol{b} \qquad k = c - \boldsymbol{\xi}^T\boldsymbol{b}$$

$$\boldsymbol{M}^{-1} = \frac{1}{k}\begin{bmatrix} k\boldsymbol{A}^{-1}+\boldsymbol{\xi}\otimes\boldsymbol{\xi}^T & -\boldsymbol{\xi} \\ -\boldsymbol{\xi}^T & 1 \end{bmatrix}$$

where: $\boldsymbol{\xi}\otimes\boldsymbol{\xi}^T$ means the tensor multiplication of vectors and the result is a matrix.

All computations needed are of $O(N^2)$ computational complexity. It means that an inverse matrix can be computed incrementally with $O(N^2)$ complexity instead of $O(N^3)$ complexity required originally in this specific case. The structure of the matrix $\boldsymbol{M}$ is "similar" to the matrix of the RBF specification. The matrix $\mathbf{A}$ in the equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is symmetrical and non-singular if appropriate rules for RBFs are kept.

Now, the question is how the incremental computation of an inverse matrix can be used for RBF interpolation?

A. Point Insertion

Let us assume a simple situation when the interpolation for N points has been computed and we need to include a new point into the given data set. A brute force approach of full RBF computation on the new data set can be used with $O(N^3)$ complexity computation.

If the RBF interpolation for $N+1$ points is considered, the following system of equations is obtained:

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,N} & \varphi_{1,N+1} & x_1 & y_1 & 1 \\ \vdots & \ddots & \cdots & \vdots & \vdots & \vdots & 1 \\ \varphi_{N,1} & \vdots & \varphi_{N,N} & \varphi_{N,N+1} & x_N & y_N & 1 \\ \varphi_{N+1,1} & & \varphi_{N+1,N} & \varphi_{N+1,N+1} & x_{N+1} & y_{N+1} & 1 \\ x_1 & \cdots & x_N & x_{N+1} & 0 & 0 & 0 \\ y_1 & \cdots & y_N & y_{N+1} & 0 & 0 & 0 \\ 1 & \cdots & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \lambda_{N+1} \\ a_x \\ a_y \\ a_0 \end{bmatrix} = [h_1 \quad \cdots \quad h_N \quad h_{N+1} \quad 0 \quad 0 \quad 0]^T$$

where: $\varphi_{i,j} = \varphi_{j,i}$. Reordering the equations above we get:

$$\begin{bmatrix} 0 & 0 & 0 & x_1 & \cdots & x_N & x_{N+1} \\ 0 & 0 & 0 & y_1 & \cdots & y_N & y_{N+1} \\ 0 & 0 & 0 & 1 & \cdots & 1 & 1 \\ x_1 & y_1 & 1 & \varphi_{1,1} & \cdots & \varphi_{1,N} & \varphi_{1,N+1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_N & y_N & 1 & \varphi_{N,1} & \cdots & \varphi_{N,N} & \varphi_{N,N+1} \\ x_{N+1} & y_{N+1} & 1 & \varphi_{N+1,1} & \cdots & \varphi_{N+1,N} & \varphi_{N+1,N+1} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_0 \\ \lambda_1 \\ \vdots \\ \lambda_N \\ \lambda_{N+1} \end{bmatrix} = [0 \quad 0 \quad 0 \quad h_1 \quad \cdots \quad h_N \quad h_{N+1}]^T$$

The last row and the last column is "inserted". As RBF functions are symmetrical, the recently derived formula for iterative computation of the inverse function can be used directly. The RBF interpolation is given by the matrix $\boldsymbol{M}$ as:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

where the matrix $\boldsymbol{A}$ is the RBF $(N+3) \times (N+3)$ matrix and the $(N+3)$ vector $\boldsymbol{b}$ and scalar value c are defined as:

$$\boldsymbol{b} = [x_{N+1} \quad y_{N+1} \quad 1 \quad \varphi_{1,N+1} \quad \cdots \quad \varphi_{N,N+1}]^T$$
$$c = \varphi_{N+1,N+1}$$

 ISBN: 978-1-61208-184-7

It means that it is possible to compute the *(N+1) × (N+1)* matrix $\boldsymbol{M}^{-1}$ if the $N \times N$ matrix $\boldsymbol{A}^{-1}$ is known with $O(N^2)$ complexity.

That is exactly what we wanted!

Now we have proved that the iterative computation of inverse function is of $O(N^2)$complexity offers a significant performance improvement for points insertion. It should be noted that some operations can be implemented more effectively, especially $\boldsymbol{\xi}\otimes\boldsymbol{\xi^T} = \boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{A}^{-1}$ as the matrix $\boldsymbol{A}^{-1}$ is symmetrical etc.

B. Point removal

In some cases it is necessary to remove a point from the given data set. It is actually an inverse operation to the insertion operation described above. Let us consider a matrix $\boldsymbol{M}$ of the size *(N+1) × (N+1)* as

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

Now, the inverse matrix $\boldsymbol{M}^{-1}$ is known and we want to compute matrix $\boldsymbol{A}^{-1}$, which is of the size $N \times N$.

Recently, derived opposite rule:

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^T & c \end{bmatrix}$$

$$\boldsymbol{\xi} = \boldsymbol{A}^{-1}\boldsymbol{b} \qquad k = c - \boldsymbol{\xi^T}\boldsymbol{b}$$

$$\boldsymbol{M}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi^T} & -\frac{1}{k}\boldsymbol{\xi} \\ -\frac{1}{k}\boldsymbol{\xi^T} & \frac{1}{k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix}$$

It can be seen that:

$$\boldsymbol{Q}_{11} = \boldsymbol{A}^{-1} + \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi^T}$$

and, therefore,:

$$\boldsymbol{A}^{-1} = \boldsymbol{Q}_{11} - \frac{1}{k}\boldsymbol{\xi}\otimes\boldsymbol{\xi^T}$$

Now there are known both operations, i.e. insertion and removal, with effective computation of $O(N^2)$ computational complexity instead of $O(N^3)$. It should be noted that vectors related to the point assigned for a removal must be in the last row and last column of the matrix $\boldsymbol{M}^{-1}$.

C. Point selection

As the number of points within a given data set could be high, the point removal might be driven by a requirement of removing a point causing a *minimal interpolation error*. This is a tricky requirement as there is probably no general answer. The requirement should include additional information which interval of $\boldsymbol{x}$ is to be considered.

Generally, we have a function:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \lambda_j \varphi_j(\boldsymbol{x}) + P_k(\boldsymbol{x})$$

And we want to remove a point $\boldsymbol{x}_j$ which causes a minimal interpolation error ε_j, i.e.

$$f_i(\boldsymbol{x}) = \sum_{j=1, i\neq j}^{N} \lambda_j \varphi_j(\boldsymbol{x}) + P_k(\boldsymbol{x})$$

and the following should be minimized:

$$\varepsilon_i = \int_{\Omega} \left| f(\boldsymbol{x}) - f_j(\boldsymbol{x}) \right| d\boldsymbol{x}$$

where: Ω is the interval on which the interpolation is to be made. It means that if the point $\boldsymbol{x}_j$ is removed the error ε_i is determined as:

$$\varepsilon_i = \lambda_i \int_{\Omega} |\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)| d\boldsymbol{x}$$

As the interval Ω on which the interpolation is known, we can compute or estimate the error ε_j for each point $\boldsymbol{x}_j$ in the given data set and select the best one. For many functions φ, the error ε_j can be computed or estimated analytically as the evaluation of ε_j is simple, e.g.

$$\int r^m \ln \mathrm{r}\ dr = r^{m+1}\left[\frac{ln\, r}{m+1} - \frac{1}{(m+1)^2}\right] \quad m \neq -1$$

In particular, it means that for TPS function $r^2 \ln r$ the error ε_k is easy to evaluate. In the case of CSRBFs, the estimation is even simpler as they have a limited influence, so generally λ_j determines the error ε_j.

It should be noted that a selection of a point with the lowest influence to the interpolation precision in the given interval Ω is of $O(N)$ complexity only.

The above has shown a new approach to RBF computation which is convenient for larger data sets. It is especially convenient for t-varying data and for applications, where a "sliding window" needs to be used. Additionally basic operations – point insertion and point removal – have been introduced. These operations have $O(N^2)$ computational complexity only, which makes a significant difference from the original approach used for RBFs computation having $O(N^3)$.

IV. SCATTERED DATA RBF INTERPOLATION

The RBF interpolation relies on solution of a LSE $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ of the size $N \times N$ in principle, where N is a number of the data processed. If the "global" functions are used, the matrix $\boldsymbol{A}$ is full, while if the "local" functions are used (CSRBF), the matrix $\boldsymbol{A}$ is sparse.

However, in visualization applications it is necessary to compute the final function $f(\boldsymbol{x})$ many many times and even for already computed λ_i values, the computation of $f(\boldsymbol{x})$ is too expensive. Therefore it is reasonable to significantly "reduce" the dimensionality of the LSE $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Of course, we are now changing the interpolation property of the RBF to approximation, i.e. the values computed do not pass the given values exactly.

Probably the best way is to formulate the problem using the Least Square Error approximation. Let us consider the formulation of the RBF interpolation again.

 ISBN: 978-1-61208-184-7

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{M} \lambda_j \, \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\right\|\right) + \boldsymbol{a}^T \boldsymbol{x}_i + a_0$$

$$h_i = f(\boldsymbol{x}_i) \qquad i = 1, \ldots, N$$

where: $\boldsymbol{\xi}_j$ are not given points, but points in a pre-defined "virtual mesh" as only coordinates are needed (there is no tessellation needed). This "virtual mesh" can be irregular, orthogonal, regular, adaptive etc. For simplicity, let us consider 2-dimensional squared (orthogonal) mesh in the following example. Then the $\boldsymbol{\xi}_j$ coordinates are the corners of this mesh. It means that the given scattered data will be actually "re-sampled", e.g. to the squared mesh.

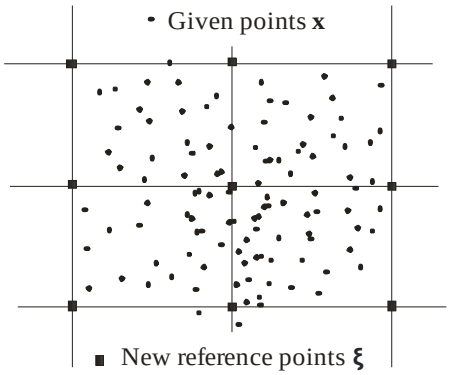


Figure 2. RBF approximation and points' reduction

In many applications the given data sets are heavily over sampled, or for the fast previews, e.g. for the WEB applications, we can afford to "down sample" the given data set. Therefore the question is how to reduce the resulting size of LSE.

Let us consider that for the visualization purposes we want to represent the final potential field in d-dimensional space by M values instead of N and $M \ll N$. The reason is very simple as if we need to compute the function $f(\boldsymbol{x})$ in many points, the formula above needs to be evaluated many times. We can expect that the number of evaluation Q can be easily requested at $10^2\,N$ of points (new points) used for visualization.

If we consider that $Q \geq 10^2\,N$ and $N \geq 10^2\,M$ then **the speed up factor in evaluation can be easily about 10^4** !

This formulation leads to a solution of a linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ where number of rows $N \gg M$, number of unknown $[\lambda_1, \ldots, \lambda_M]^T$. As the application of RBF is targeted to high dimensional visualization, it should be noted that the polynomial is not requested for all kernels of the RBF interpolation. But it is needed for $\varphi(r) = r^2 lg\, r$ kernel function (TPS). This reduces the size of the linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ significantly and can be solved by the Least Square Method (LSM) as $\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b}$ or Singular Value Decomposition (SVD) can be used.

$$\begin{bmatrix} \varphi_{1,1} & \cdots & \varphi_{1,M} \\ \vdots & \ddots & \vdots \\ \varphi_{i,1} & \cdots & \varphi_{i,M} \\ \vdots & \ddots & \vdots \\ \varphi_{N,1} & \cdots & \varphi_{N,M} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ \vdots \\ \vdots \\ h_N \end{bmatrix} \qquad \boldsymbol{Ax} = \boldsymbol{b}$$

The high dimensional data can be approximated for visualization by RBF efficiently with a high flexibility as it is possible to add additional points of an area of interest to the mesh. It means that a user can add some points to already given mesh and represent easily some details if requested. It should be noted that the use of LSM increases instability of the LSE in general.

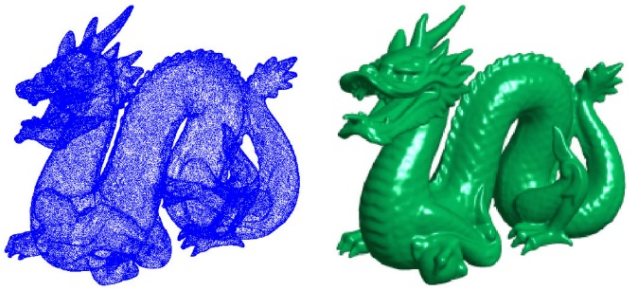
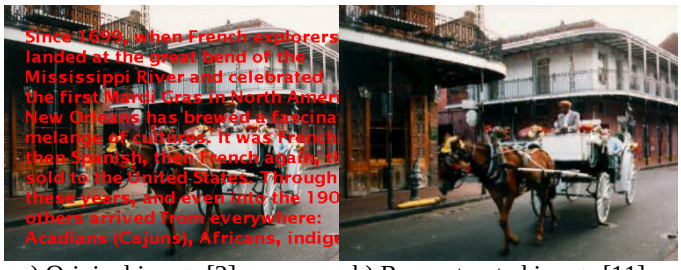


Figure 3. Surface reconstruction (438 000 points) [3]

Experimental evaluation

The RBF interpolation is a very powerful tool for interpolation of data in d-dimensional space in general. In order to demonstrate the functionality the RBF, we have recently used RBF for reconstruction of damaged images by a noise or by inpainting. Also a surface reconstruction has been solved by the RBF interpolation well. Fig. 3–5 illustrate the power of the RBF interpolation [2][3][8][15].



a) Original image [2] b) Reconstructed image [11]

Figure 4. Inpaited image reconstruction

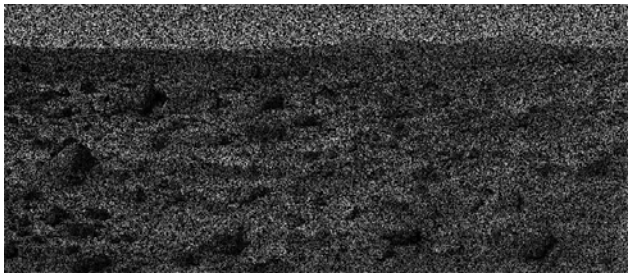


Figure 5a. Original image with 60% of damaged pixels [11]

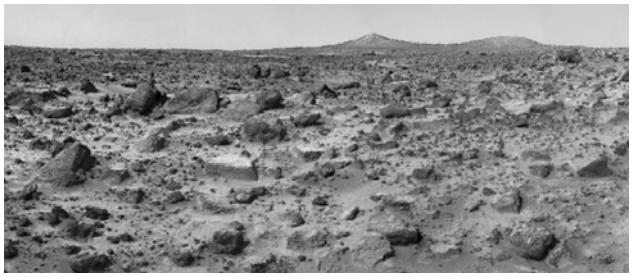


Figure 5b. Reconstructed image [11]

The RBF interpolation gives quite good results even if the images are heavily damaged. The advantages of RBF interpolation over the other interpolations have been proved even though that the RBF interpolation causes some additional computational cost as the RBF is primarily targeted for scattered data interpolation.

V. CONCLUSION

The radial basis functions (RBF) interpolation is a representative interpolation method for unordered scattered data sets. It is well suited approach for solving problems without meshing the data domain. RBF interpolations are used in many computational fields, e.g in solution of partial differential equations etc. RBF approach supports interpolation in the *d*-dimensional space naturally.

This paper describes an incremental computation of RBF and shows the decrease of the computational complexity from approx. $O(N^3)$ to $O(N^2)$ for a point insertion and a point removal.

It also presents a method for "resampling" the data processed as the approximation is acceptable in many applications, namely in visualization. The approach enables to increase details for visualization by adding new points to the "virtual mesh", if more details are needed. It is necessary to mention, that there is no mesh actually needed and only points of the "virtual mesh" need to be defined.

Future research should be devoted to the evaluation of computing precision and stability as the RBF interpolation generally leads to not well conditioned LSE. Also, there is a need to analyze, how the ratio $\nu = N/M$ can be controlled effectively and what can be expected in real and large data applications, e.g. from GIS fields, inverse engineering process in CAD/CAM etc.

ACKNOWLEDGMENT

The author thanks to colleagues at the University of West Bohemia (UWB) in Plzen and at the VSB-Technical University in Ostrava for their critical comments and constructive suggestions, to anonymous reviewers for their critical view and comments that helped to improve the manuscript. Special thanks belong to RongJiang Pan, Shandong University, China for recommendations during his stay at the University of West Bohemia (UWB), to former PhD and MSc. students at the UWB Vit Ondracka, Lukas Loukota, Jan Hobza, Karel Uhlir and Jiri Zapletal.

This research was supported by the Ministry of Education of the Czech Republic, projects ME10060 and LA10035.

REFERENCES

[1] B.J.Ch. Baxter, "The Interpolation Theory of Radial Basis Functions," PhD thesis, Trinity College, Cambridge University, U.K., 1992.

[2] M. Bertalmio, G. Sapiro, C. Ballester and V. Caselles, "Image Inpainting," Proceedings of SIGGRAPH'00, Computer Graphics, pp. 417-424, 2000.

[3] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum and T.R. Evans, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," Computer Graphics (SIGGRAPH 2001 proceedings), pp. 67-76, 2001.

[4] J. Duchon, "Splines Minimizing Rotation-invariant Semi-norms in Sobolev space," in Constructive Theory of Functions of Several Variables, Springer Lecture Notes in Math, Vol. 21, pp. 85-100, 1977.

[5] L.R. Hardy, "Multiquadric Equations of Topography and other Irregular Surfaces", J. Geophysical. Res., Vol. 76, pp. 1905-1915, 1971.

[6] C.A. Micchelli, "Interpolation of Scattered Data: Distance Matrix and Conditionally Positive Definite Functions," Constr. Approx., No. 2, pp. 11-22, 1986.

[7] R. Pan and V. Skala, "Implicit Surface Modeling Suitable for Inside/Outside Tests with Radial Basis Functions," 10th International Conference on Computer Aided Design and Computer Graphics (CAD/Graphics), 2007.

[8] R. Pan and V. Skala, "A Two-Level Approach to Implicit Modeling with Compactly Supported Radial Basis Functions," Engineering and Computers, Springer Verlag, Vol. 27. No. 3., pp. 299-307, ISSN 0177-0667, 2011.

[9] R. Pan and V. Skala, "Continuous Global Optimization in Surface Reconstruction from an Oriented Point Cloud," Computer Aided Design, Vol. 43, No. 8, pp. 896-901, Elsevier, 2011.

[10] I.P. Schagen, "Interpolation in Two Dimension – A New Technique," IMA Journal of Applied Mathematics, Vol. 23, No. 1, pp. 53-59, 1977.

[11] K.Uhlir and V. Skala, "Radial Basis Function use for the Restoration of Damaged Images," in Computer Vision and Graphics, Dordrecht: Springer, pp. 839-844, 2006.

[12] Ch.C.L. Wang and T.-H. Kwok, "Interactive Image Inpainting using DCT Based Exemplar Matching," ISVC 2009, LNCS 5876, pp. 709-718, 2009.

[13] H. Wendland, "Computational Aspects of Radial Basis Function Approximation," in Topics in Multivariate Approximation and Interpolation (Ed.K. Jetter et al.), Elsevier B.V., pp. 231-256, 2005.

[14] G.B. Wright, "Radial Basis Function Interpolation: Numerical and Analytical Developments," University of Colorado, PhD Thesis, 2003.

[15] J. Zapletal, P. Vanecek and V. Skala, "RBF-based Image Restoration Utilising Auxiliary Points," CGI 2009 proceedings, ACM, pp. 39-44, 2009.

[16] Y. Ohtake, A. Belyaev and H.-P. Seidel, "3D Scattered Data Interpolation and Approximation with Multilevel Compactly Supported RBFs," Graphical Models, Vol. 67, No. 3, pp. 150-165, 2005.

WEB references

FastRBF: http://www.farfieldtechnology.com/. <retrieved: 2011-12-05>

 ISBN: 978-1-61208-184-7